Chapter 3 – Binary Number System

The binary number system, also known as the base-2 numeral system, is a method of representing numbers that uses only two symbols: typically, 0 and 1. This system is the foundation of all modern computing and digital systems. Here's a breakdown of its key characteristics:

1. **Base-2 System:** Unlike the decimal system which is base-10 and uses ten digits (0-9), the binary system is base-2 and uses only two digits, 0 and 1. Each digit in a binary number is known as a bit.

2. **Positional Value**: In binary, the value of each position in a number increases exponentially to the base of 2 as you move from right to left. The rightmost position represents 2^0, the next represents 2^1, then 2^2, and so on.

3. **Representation of Numbers**: In binary, numbers are represented by a combination of 0s and 1s. For example, the binary number 1011 represents the decimal number 11. This is calculated as $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$, which is equals to $8 + 0 + 2 + 1 = 11$.

4. **Application in Computers**: Computers use the binary number system due to its simple and efficient representation using two states, which can be easily implemented with electronic circuitry using switches. These switches can be in one of two states, representing 0 or 1.

5. **Binary in Digital Data**: All types of digital data, including text, images, and sound, are ultimately represented in binary format in computer systems. This binary data is processed, stored, and transmitted by computers.

6. **Logical Operations**: Binary numbers are also fundamental in performing logical operations in computing, where binary logic forms the basis of decision-making processes in computer algorithms.

Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Binary Addition Exercise

Instructions: Add the following binary numbers. Remember, in binary addition, 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 10 (where '1' carries over to the next higher bit).

1. 1010 + 1101
2. 111 + 101
3. 1001 + 100
4. 11011 + 1011
5. 11101 + 111

Answers:

1. 10111
2. 1100
3. 1101
4. 100110
5. 110100

Binary Multiplication Exercise

Instructions: Multiply the following binary numbers. Remember, binary multiplication is similar to decimal multiplication; 0 multiplied by anything gives 0, and 1 multiplied by anything gives the number itself.

1. 101 x 11
2. 110 x 10
3. 111 x 101
4. 1001 x 110
5. 1010 x 101

Answers:

1. 1111
2. 1100
3. 100011
4. 1101110
5. 101010

ASCII, which stands for American Standard Code for Information Interchange, is a character encoding standard used for representing text in computers and other devices that use text. It's one of the most fundamental methods for encoding characters into numerical form, particularly in early computing, and still has relevance today. Here are the key aspects of ASCII:

American Standard Code for Information Interchange (ASCII)

$b_4b_3b_2b_1$	$b_7b_6b_5$ = 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	¦
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DEL

Control characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

1. **Basic Concept**: ASCII assigns a unique number to each letter, digit, and symbol. For example, the uppercase letter 'A' is represented by the number 65, the lowercase letter 'a' is 97, and the number '1' is 49.

2. **Binary Representation:** In ASCII, each character is represented by a 7-bit binary number. Since there are 7 bits, there are 128 possible combinations (2^7), allowing for 128 unique characters.

3. **Character Set:** The ASCII character set includes:
- Uppercase and lowercase English letters (A-Z, a-z)
- Digits (0-9)
- Punctuation marks (like ., ;, !, ?)
- Special characters (like @, #, $, %)
- Control characters (like newline, carriage return, tab)

4. **Control Characters:** ASCII includes control characters that don't represent printable characters but serve as commands for controlling devices such as printers or display screens. For instance, ASCII 10 represents the 'Line Feed' (new line) control character.

5. **Extended ASCII:** While standard ASCII uses 7 bits, Extended ASCII uses 8 bits (1 byte), allowing for 256 (2^8) characters. This extension was used to include additional characters, like accented letters and other symbols, catering to a broader range of applications.

6. **Importance in Computing:**

    a. **Compatibility:** ASCII was one of the first widely adopted encoding standards, ensuring compatibility across different computers and systems.
    b. Foundation for Other Encodings: ASCII forms the basis for several other character encoding systems, such as UTF-8 in Unicode, which includes ASCII as a subset.
    c. **Data Transmission:** ASCII is used in data transmission protocols, as many control characters are crucial for communication standards.
    d.

7. **Limitations**: The main limitation of ASCII is its inability to represent characters from languages other than English or special symbols not included in the 128 or 256 character set. This limitation led to the development of other encoding systems like Unicode, which can represent thousands of different characters and symbols.

Binary logic, also known as Boolean logic, is a fundamental concept in computer science and digital electronics. It refers to a form of algebra that focuses on the manipulation of binary values, typically represented as 1 (true) and 0 (false). Here are the key aspects of binary logic:

1. **Binary Values**: In binary logic, there are only two possible values for any variable: 1 (true) or 0 (false). These values are used to represent the state of a digital system, such as on/off, high/low, or yes/no.

2. **Logical Operations**: Binary logic is characterized by specific operations that can be performed on binary values. The most common operations are:

- AND: Yields true (1) only if both operands are true. For example, 1 AND 1 = 1, but 1 AND 0 = 0.
- OR: Yields true if at least one of the operands is true. For example, 1 OR 0 = 1.
- NOT: Inverts the value, turning true into false and vice versa. For example, NOT 1 = 0.

3. **Boolean Algebra**: Developed by George Boole, Boolean algebra is the theoretical basis of binary logic. It deals with variables that can have two values and the logical operations that can be applied to these variables.

4. **Truth Tables**: Binary logic can be represented using truth tables, which list the output of a logic operation for all possible combinations of inputs. For example, a truth table for the AND operation would show the result of A AND B for all combinations of A and B being 0 or 1.

5. **Applications in Digital Circuits**: Binary logic is the foundation of digital circuit design. Logic gates, which are the basic building blocks of digital circuits, operate based on binary logic. For example, an AND gate outputs a 1 only when all its inputs are 1.

6. **Computer Programming**: In computer programming, binary logic is used in decision-making structures and conditional logic. It forms the basis of operations that involve comparisons, decision-making, and controlling program flow.

7. **Data Representation and Processing**: Computers use binary logic to process and represent data. Every operation, from simple arithmetic to complex algorithmic processing, is ultimately broken down into binary logic operations at the hardware level.